INSTITUTO FEDERAL DE SANTA CATARINA

CÂMPUS FLORIANÓPOLIS

DEPARTAMENTO ACADÊMICO DE ELETRÔNICA

Cleisson Fernandes da Silva

**CONTROLADOR DIGITAL UTILIZANDO O MÉTODO DO LUGAR DAS RAÍZES**

Florianópolis, 2018

**SUMÁRIO**

[1 INTRODUÇÃO 3](#_Toc528773021)

[2 CONTROLADOR DIGITAL 4](#_Toc528773022)

[2.1 A PLANTA 4](#_Toc528773023)

[2.2 FUNÇÃO DE TRANSFERÊNCIA EXPERIMENTAL 4](#_Toc528773024)

[2.3 FUNÇÃO DE TRANSFERÊNCIA TEÓRICA 7](#_Toc528773025)

[2.4 PROJETO DO CONTROLADOR 8](#_Toc528773026)

[2.5 SIMULAÇÃO 11](#_Toc528773027)

[2.6 IMPLEMENTAÇÃO 15](#_Toc528773028)

[2.7 COMPARAÇÃO 17](#_Toc528773029)

[3 CONCLUSÃO 18](#_Toc528773030)

[APÊNDICE A – Códigos Matlab 19](#_Toc528773031)

[APÊNDICE B – Códigos Microcontrolador 20](#_Toc528773032)

[BIBLIOGRAFIA 21](#_Toc528773033)

# INTRODUÇÃO

O uso de controladores digitais está cada vez maior. Isso se deve ao fato que controladores digitais são confiáveis, baratos e são capazes de assumir as funções de um controlador analógico. Isso é feito em grande parte porque é econômico, já que o hardware é substituído por software. Funções complexas também são muito mais fáceis de implementar e o registro e o monitoramento são fáceis com computadores. Os controladores digitais também são capazes de assumir muitas formas, como a de uma placa de microprocessador, um microcontrolador ou um controlador lógico programável. Os sistemas de controle digital são flexíveis, baratos, escaláveis ​​e adaptáveis. Portanto, eles são amplamente utilizados em implementações de múltiplos sistemas de controle.

# CONTROLADOR DIGITAL

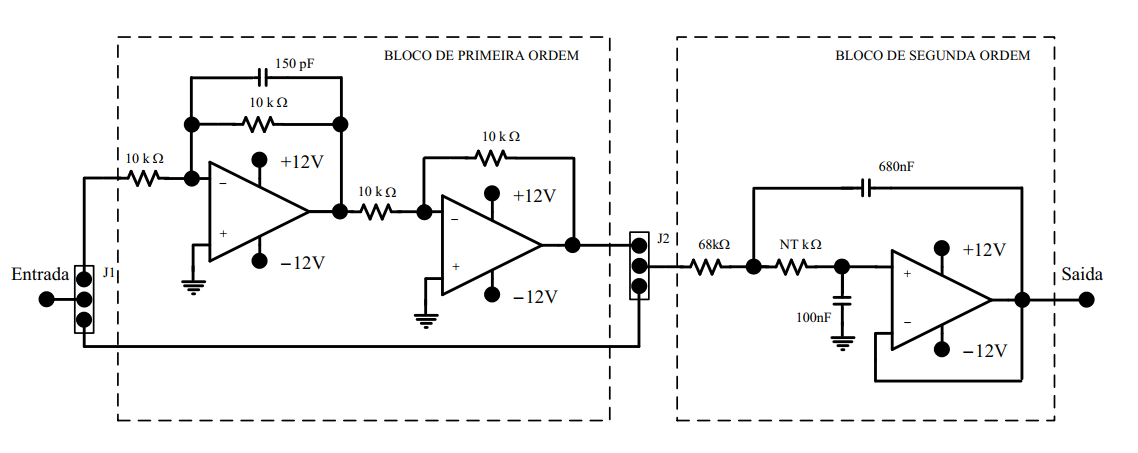
O controle digital é um ramo da teoria de controle que usa computadores digitais para atuar como controladores de uma planta. Dependendo dos requisitos, um sistema de controle digital pode assumir a forma de um microcontrolador ou um ASIC ou um computador de mesa. Como um computador digital é um sistema discreto, a transformada de Laplace é substituída pela transformada Z. Além disso, como um computador digital tem precisão finita, é necessário cuidado extra para garantir que o erro nos coeficientes, conversão A / D, conversão D / A, etc. não esteja produzindo efeitos indesejados ou não planejados.

A aplicação do controle digital pode ser facilmente compreendido no uso de feedback. Desde a criação do primeiro computador digital no início da década de 1940, o preço dos computadores digitais caiu consideravelmente, o que os tornou peças-chave para controlar os sistemas por várias razões.

## A PLANTA

A planta a ser controlada está apresentada na Figura 1, onde NT é 24. Além disso o capacitor de 150 pF foi substituído por um capacitor de 15 nF. Essa planta é um circuito que contém um bloco de primeira ordem e um bloco de segunda ordem em sequência. A função de transferência desta planta será obtida experimentalmente e analiticamente.

Figura 1 – Circuito da Planta a ser controlada



## FUNÇÃO DE TRANSFERÊNCIA EXPERIMENTAL

Utilizando as capturas do osciloscópio e obtendo a saída da planta com a entrada de um degrau, podemos obter uma função de transferência de cada bloco que compõe o sistema. As capturas obtidas estão apresentadas na Figura 3, Figura 4 e Figura 5.

Figura 2 – Captura do tempo de acomodação do bloco 1

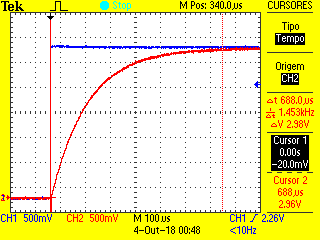


Figura 3 – Captura do tempo de pico do bloco 2

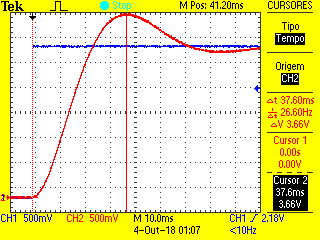


Figura 4 – Captura da variação inferior do sobressinal do bloco 2

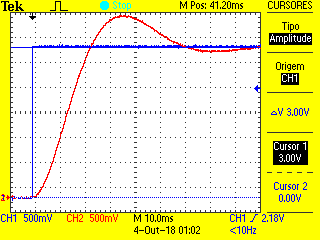
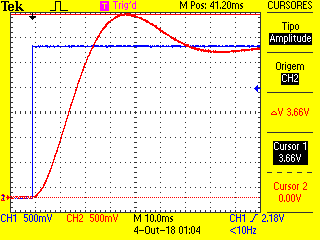


Figura 5 – Captura da variação superior do sobressinal do bloco 2



Para o sistema de primeira ordem temos:

Considerando que o sistema de primeira ordem chega em regime permanente com 5 constantes de tempo, podemos encontrar a função de transferência deste bloco:

Para o sistema de segunda ordem temos:

Com os valores de máximo de sobressinal e o tempo de pico pode-se encontrar:

Com estes parâmetros encontramos a seguinte função de transferência do segundo bloco:

Assim, a função de transferência total da planta é:

## FUNÇÃO DE TRANSFERÊNCIA TEÓRICA

Modelando matematicamente o sistema obtemos a seguinte função de transferência para o bloco de primeira ordem:

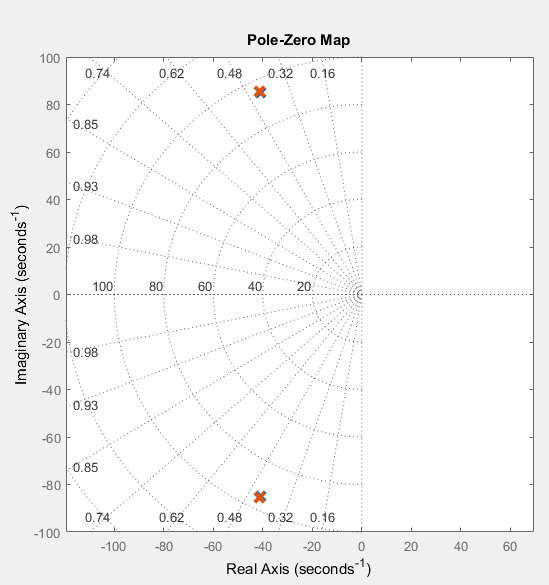
O bloco de segunda ordem é um filtro passa baixa com topologia Sallen-key e tem a seguinte função de transferência para o bloco:

Utilizando-se NT = 24, temos:

Portanto a função de transferência da planta completa se torna:

Os polos dominantes desta função de transferência estão em -41.3963 ± 85.4255i. enquanto que os polos dominantes da função de transferência prática estão em -40.9224 ± 84.9079i. estes polos estão plotados na Figura 6 para uma melhor análise.

Figura 6 – Localização dos polos dominantes



## PROJETO DO CONTROLADOR

Os requisitos de projeto são:

* Reduzir o sobressinal em pelo menos 50% do valor obtido em malha aberta
* Reduzir o tempo de acomodação em pelo menos 50% do valor obtido em malha aberta
* Erro de regime permanente ao degrau nulo

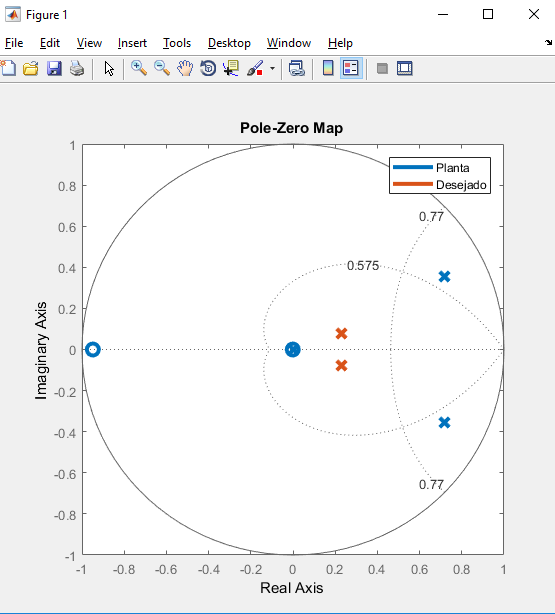
Para reduzir o sobressinal em 50% precisamos de um valor de . O sistema original possuía um tempo de acomodação de Para reduzirmos em 50% o tempo de acomodação do sistema original prático, para menor que 36.7 ms, é preciso .

A taxa de amostragem é obtida pelo critério de dez amostras por ciclo da oscilação (OGATA, 1995).

A função de transferência discreta do sistema prático pelo método *zero order hold* é expressa como:

Os polos dominantes do sistema devem estar na região em que satisfaz as duas condições. A partir disso foi escolhido o ponto do plano z para ser parte do lugar das raízes como sendo o ponto , mostrado na Figura 7.

Figura 7 – Localização dos polos desejado



Para que o ponto z escolhido seja parte do lugar das raízes, este deve satisfazer a condição de ângulo e portanto a soma dos ângulos no ponto z deve ser igual à . Calculando a função de transferência discreta para o ponto z resulta em um ângulo complexo de . Portanto o controlador deve fornecer .

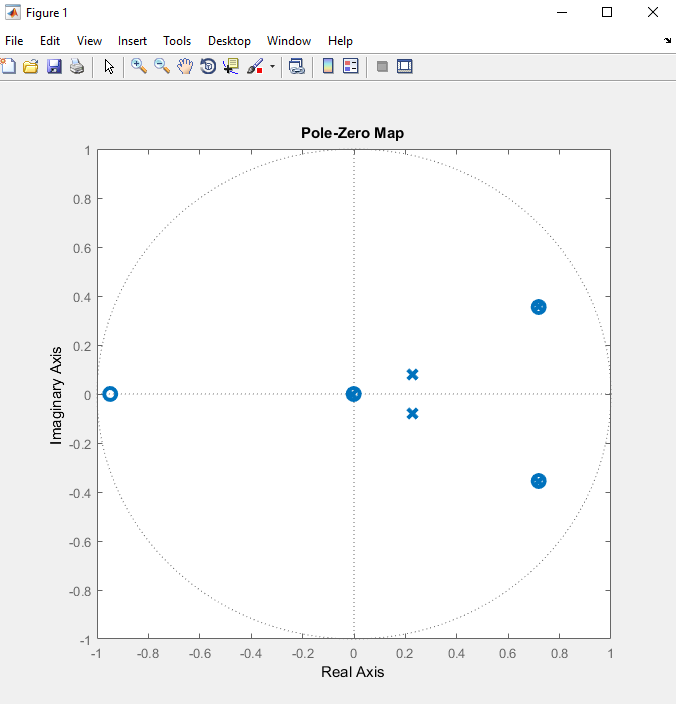
Para remover o erro de regime permanente ao degrau é necessário que o controlador contenha um polo em 1. Para o controlador foi considerado anular os polos complexos da função de transferência discreta da planta com zeros complexos e encontrar um polo que satisfaça a condição de ângulo do lugar das raízes. Portanto a função de transferência pulsada do controlador tem a seguinte forma:

Encontra-se que o polo deve ter um ângulo de 9.7203°. Assim encontramos que o valor deste polo deve ser z = -0.2390. O função de transferência do controlador se torna a seguinte:

Para determinarmos K precisamos satisfazer a condição de módulo do lugar das raízes e encontramos que K = 2.9506. Portanto a equação de transferência discreta do compensador resulta em:

Os polos e zeros de malha fechada do sistema final estão apresentados na Figura 8.

Figura 8 – Polos e zeros em malha fechada



O ponto pode ser considerado o polo dominante da função de transferência de malha fechada devido aos cancelamentos de polos e zeros. Portanto, foi calculado os valores das especificações com base nesse ponto. Foi obtido os seguintes valores para o ponto z:

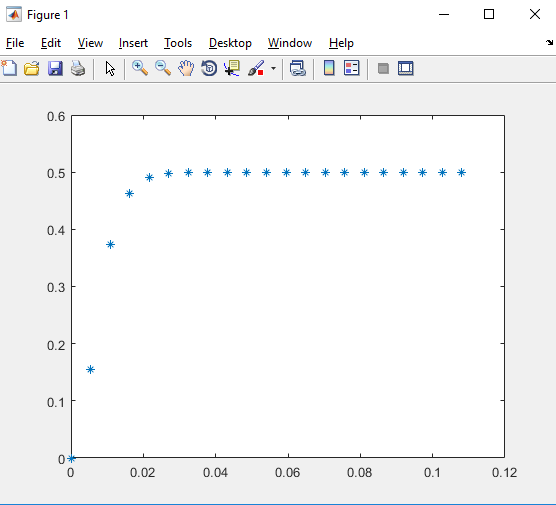
Com esses valores conseguimos obter as especificações:

Utilizando o teorema do valor final, verificamos o erro de regime permanente para a entrada ao degrau e à rampa do sistema final:

## SIMULAÇÃO

Na Figura 9 temos a simulação do sistema utilizando a equações de diferenças de cada bloco com uma entrada à um degrau de 0.5. O código se encontra no Apêndice A.

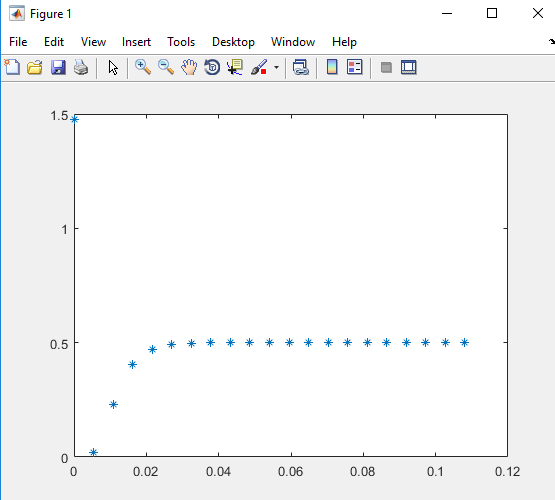
Figura 9 – Simulação implementando a equações de diferenças



As especificações que obtemos da simulação da Figura 9 são:

Um fator importante para se considerar antes de o controlador ser implementado é o esforço do controlador, ou seja, a saída de tensão que o microcontrolador fornecerá para a planta. Esse nível de tensão não pode ultrapassar o nível máximo de tensão que no caso é o mesmo nível da alimentação de 3.3V do microcontrolador e nem o nível mínimo de 0V. Observando na Figura 10, onde temos uma entrada degrau de amplitude 0.5, podemos perceber que o valor de tensão de saída do microcontrolador não vai ultrapassar os limites para os valores de *setpoints* estabelecidos e portanto este controlador pode ser implementado no microcontrolador.

Figura 10 – Ação de controle na simulação



Foi usado o Simulink para simular o comportamento do circuito completo ao se empregar a realimentação e ao se aplicar o controlador no circuito. O esquemático da simulação para medir a saída total se encontra na Figura 11. Como o Simulink adiciona implicitamente o bloco de Segurador de Ordem Zero na saída do controlador, não é necessário adicionar este bloco ao Esquemático da simulação. Assim como será implementado no microcontrolador, o bloco Repeating Sequence Stair1 alterna o valor de *setpoint* da tensão entre 1V e 1.5V. O resultado do sinal da saída está mostrado na Figura 12.

Figura 11 – Esquemático da simulação para medir a saída

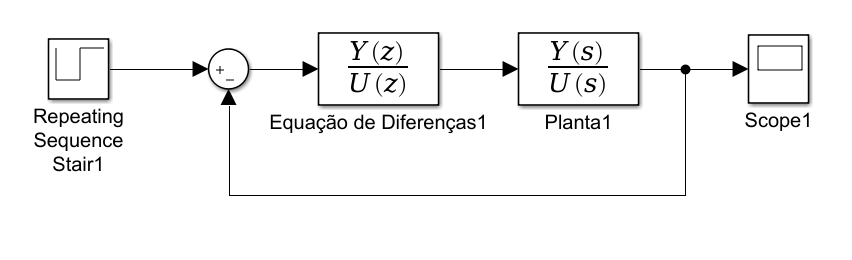


Figura 12 – Captura do sinal na saída

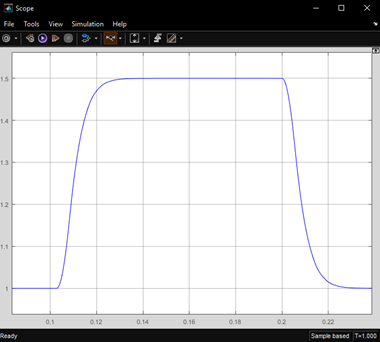


Figura 13 – Esquemático da simulação para medir o esforço do controlador

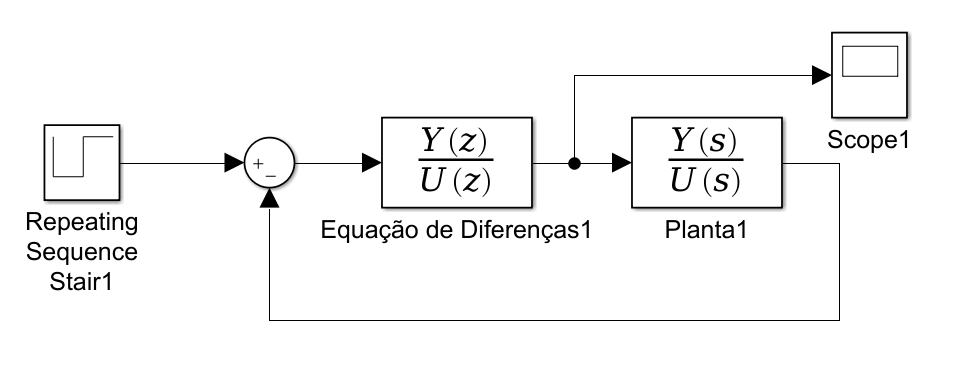
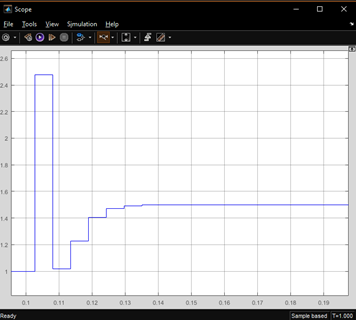


Figura 14 – Captura do sinal mostrando a saída do controlador



## IMPLEMENTAÇÃO

O controlador digital foi implementado no microcontrolador STM32F103C8T6 da STMicroeletronics. Como o microcontrolador não possui saída D/A, foi utilizado um PWM com uma tensão média correspondendo ao sinal desejado. O código utilizado se encontra no Apêndice B. Pela Figura 17 obtemos o valor de . Pela Figura 16 temos que o tempo de acomodação do sistema final é de cerca de 19 ms. Pelas capturas e valores obtidos podemos afirmar que o controlador projetado atende os requisitos de projeto.

Figura 15 – Captura do valor final do sistema

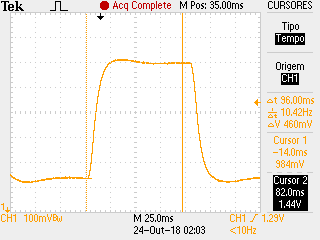


Figura 16 – Captura do tempo de acomodação do sistema

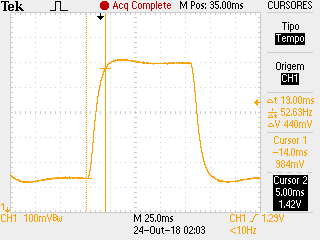
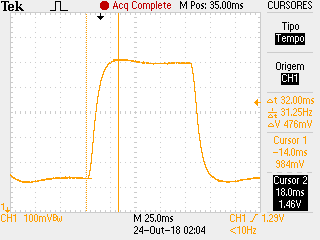
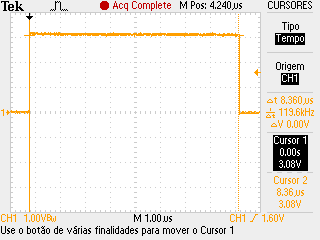


Figura 17 – Captura do pico do sistema



Na Figura 18 temos o tempo que o microcontrolador leva para processar e colocar o valor calculado na saída. Podemos ver que o tempo de processamento é de 8.36 μs, o equivalente à aproximadamente 602 ciclos do microcontrolador.

Figura 18 – Tempo de processamento



## COMPARAÇÃO

|  |  |  |  |
| --- | --- | --- | --- |
|  | Projetado | Simulado | Prático |
| Mp (%) | 1.77E-04 | 1.32E-06 | 3.48 |
| ts (ms) | 11.4 | 22 | 19 |

# CONCLUSÃO

Foi implementado com sucesso um controlador digital para a planta em questão. Esta planta apresenta valores de tempo de acomodação e valor de sobressinal menor quando colocada em malha fechada com o controlador do que apenas a planta em malha aberta.

# APÊNDICE A – Códigos Matlab

Código para o cálculo da planta:

clear;

clc;

%% Especifications

s = tf('s');

% Planta Teórica

G1t = 1/(150e-6\*s+1);

G2t = 9010.83/(s^2 + 82.8763\*s + 9010.83);

Gt = minreal(G1t\*G2t);

[numt, dent] = tfdata(Gt,'v');

% Especificações da planta prática

ts1 = 688e-6;

Mp = 0.22;

tp = 37e-3;

% Bloco primeira ordem

tau = ts1/5;

G1 = 1/(tau\*s+1);

% Bloco segunda ordem

zeta = fzero(@(x) ((log(Mp)/pi) + (x/sqrt(1-x^2))), 0.5);

wn = pi/(tp\*sqrt(1-zeta^2));

G2 = (wn^2)/(s^2 + 2\*zeta\*wn\*s + wn^2);

ts2 = 3/(zeta\*wn);

% Planta Prática Contínua

G = G1\*G2;

G = minreal(G);

% % Mostrar localização dos polos dominantes

% pzmap(G,Gt);

% xlim([-50 0]);

%% Controlador

% Requisitos

Mp = Mp/2;

ts = ts2/2;

zeta = fzero(@(x) ((log(Mp)/pi) + (x/sqrt(1-x^2))), 0.5);

wn = 3/(ts\*zeta);

Fs\_required = (10\*wn\*sqrt(1-zeta^2))/(2\*pi);

Fs = 185;

Ts = 1/Fs;

z = tf('z',Ts);

z1 = 0.228 + 0.08i; % ponto z que deseja ser parte do lugar das raízes

% Planta Prática Discreta

Gz = c2d(G,Ts);

% Mostrar Requisitos

figure;

pzmap(Gz,(1/((z-z1)\*(z-z1'))));

zgrid(zeta,wn\*Ts);

legend('Planta','Desejado')

angle\_required = -(angle(evalfr(Gz,z1)) - pi);

% O controlador anula os polos complexos da planta e contém um integrador,

% portanto só precisamos encontrar o valor de um pólo para satisfazer a

% condição de ângulo

zeros\_planta = pole(Gz);

angulo\_polo = angle(z1 - zeros\_planta(1)) + angle(z1 - zeros\_planta(2)) - angle(z1 - 1) - angle\_required;

polo\_desejado = real(z1) - imag(z1)/tan(angulo\_polo);

Cz\_semK = ((z-zeros\_planta(1))\*(z-zeros\_planta(2)))/((z-polo\_desejado)\*(z-1));

K = abs(1/(evalfr(minreal(Cz\_semK\*Gz), z1)));

Cz = K\*Cz\_semK;

FTMA = minreal(Cz\*Gz);

FTMF = minreal(feedback(FTMA,1));

[numg, deng] = tfdata(G, 'v');

[numgz, dengz] = tfdata(Gz,'v');

[numc, denc] = tfdata(Cz,'v');

Código equação de diferenças:

clc;

kmax = 20;

k = 0:kmax;

r = 0:Ts:kmax\*Ts;

e = zeros(1,length(k));

c = zeros(1,length(k));

y = zeros(1,length(k));

n = 1;

y(n) = 0;

e(n) = r(n) - y(n);

c(n) = numc(1)\*e(n);

n = 2;

y(n) = numgz(2)\*c(n-1) - dengz(2)\*y(n-1);

e(n) = r(n) - y(n);

c(n) = numc(1)\*e(n) + numc(2)\*e(n-1) - denc(2)\*c(n-1);

n = 3;

y(n) = numgz(2)\*c(n-1) + numgz(3)\*c(n-2) - dengz(2)\*y(n-1) - dengz(3)\*y(n-2);

e(n) = r(n) - y(n);

c(n) = numc(1)\*e(n) + numc(2)\*e(n-1) + numc(3)\*e(n-2) - denc(2)\*c(n-1) - denc(3)\*c(n-2);

for n = 4:length(k)

y(n) = numgz(2)\*c(n-1) + numgz(3)\*c(n-2) + numgz(4)\*c(n-3) - dengz(2)\*y(n-1) - dengz(3)\*y(n-2) - dengz(4)\*y(n-3);

e(n) = r(n) - y(n);

c(n) = numc(1)\*e(n) + numc(2)\*e(n-1) + numc(3)\*e(n-2) - denc(2)\*c(n-1) - denc(3)\*c(n-2);

end

plot(k\*Ts, y, '\*');

# APÊNDICE B – Códigos Microcontrolador

Main.c:

/\* Includes ------------------------------------------------------------------\*/

#include "main.h"

#include "stm32f1xx\_hal.h"

#include <arm\_math.h>

/\* USER CODE BEGIN Includes \*/

/\* USER CODE END Includes \*/

/\* Private variables ---------------------------------------------------------\*/

ADC\_HandleTypeDef hadc1;

TIM\_HandleTypeDef htim1;

TIM\_HandleTypeDef htim2;

TIM\_HandleTypeDef htim3;

/\* USER CODE BEGIN PV \*/

/\* Private variables ---------------------------------------------------------\*/

GPIO\_PinState state;

uint32\_t counter = 0;

uint32\_t setValue = 1000;

volatile uint32\_t rawValue;

volatile int32\_t x[3];

volatile int32\_t y[3];

/\* USER CODE END PV \*/

/\* Private function prototypes -----------------------------------------------\*/

void SystemClock\_Config(void);

static void MX\_GPIO\_Init(void);

static void MX\_TIM2\_Init(void);

static void MX\_TIM1\_Init(void);

static void MX\_TIM3\_Init(void);

static void MX\_ADC1\_Init(void);

static void MX\_NVIC\_Init(void);

void HAL\_TIM\_MspPostInit(TIM\_HandleTypeDef \*htim);

/\* USER CODE BEGIN PFP \*/

/\* Private function prototypes -----------------------------------------------\*/

/\* USER CODE END PFP \*/

/\* USER CODE BEGIN 0 \*/

/\* USER CODE END 0 \*/

/\*\*

\* @brief The application entry point.

\*

\* @retval None

\*/

int main(void)

{

/\* USER CODE BEGIN 1 \*/

/\* USER CODE END 1 \*/

/\* MCU Configuration----------------------------------------------------------\*/

/\* Reset of all peripherals, Initializes the Flash interface and the Systick. \*/

HAL\_Init();

/\* USER CODE BEGIN Init \*/

/\* USER CODE END Init \*/

/\* Configure the system clock \*/

SystemClock\_Config();

/\* USER CODE BEGIN SysInit \*/

/\* USER CODE END SysInit \*/

/\* Initialize all configured peripherals \*/

MX\_GPIO\_Init();

MX\_TIM2\_Init();

MX\_TIM1\_Init();

MX\_TIM3\_Init();

MX\_ADC1\_Init();

/\* Initialize interrupts \*/

MX\_NVIC\_Init();

/\* USER CODE BEGIN 2 \*/

HAL\_TIM\_PWM\_Start(&htim2, TIM\_CHANNEL\_1);

HAL\_TIM\_Base\_Start\_IT(&htim1);

HAL\_TIM\_Base\_Start\_IT(&htim3);

HAL\_ADC\_Start\_IT(&hadc1);

/\* USER CODE END 2 \*/

/\* Infinite loop \*/

/\* USER CODE BEGIN WHILE \*/

while (1)

{

/\* USER CODE END WHILE \*/

/\* USER CODE BEGIN 3 \*/

}

/\* USER CODE END 3 \*/

}

/\*\*

\* @brief System Clock Configuration

\* @retval None

\*/

void SystemClock\_Config(void)

{

RCC\_OscInitTypeDef RCC\_OscInitStruct;

RCC\_ClkInitTypeDef RCC\_ClkInitStruct;

RCC\_PeriphCLKInitTypeDef PeriphClkInit;

/\*\*Initializes the CPU, AHB and APB busses clocks

\*/

RCC\_OscInitStruct.OscillatorType = RCC\_OSCILLATORTYPE\_HSE;

RCC\_OscInitStruct.HSEState = RCC\_HSE\_ON;

RCC\_OscInitStruct.HSEPredivValue = RCC\_HSE\_PREDIV\_DIV1;

RCC\_OscInitStruct.HSIState = RCC\_HSI\_ON;

RCC\_OscInitStruct.PLL.PLLState = RCC\_PLL\_ON;

RCC\_OscInitStruct.PLL.PLLSource = RCC\_PLLSOURCE\_HSE;

RCC\_OscInitStruct.PLL.PLLMUL = RCC\_PLL\_MUL9;

if (HAL\_RCC\_OscConfig(&RCC\_OscInitStruct) != HAL\_OK)

{

\_Error\_Handler(\_\_FILE\_\_, \_\_LINE\_\_);

}

/\*\*Initializes the CPU, AHB and APB busses clocks

\*/

RCC\_ClkInitStruct.ClockType = RCC\_CLOCKTYPE\_HCLK|RCC\_CLOCKTYPE\_SYSCLK

|RCC\_CLOCKTYPE\_PCLK1|RCC\_CLOCKTYPE\_PCLK2;

RCC\_ClkInitStruct.SYSCLKSource = RCC\_SYSCLKSOURCE\_PLLCLK;

RCC\_ClkInitStruct.AHBCLKDivider = RCC\_SYSCLK\_DIV1;

RCC\_ClkInitStruct.APB1CLKDivider = RCC\_HCLK\_DIV2;

RCC\_ClkInitStruct.APB2CLKDivider = RCC\_HCLK\_DIV1;

if (HAL\_RCC\_ClockConfig(&RCC\_ClkInitStruct, FLASH\_LATENCY\_2) != HAL\_OK)

{

\_Error\_Handler(\_\_FILE\_\_, \_\_LINE\_\_);

}

PeriphClkInit.PeriphClockSelection = RCC\_PERIPHCLK\_ADC;

PeriphClkInit.AdcClockSelection = RCC\_ADCPCLK2\_DIV6;

if (HAL\_RCCEx\_PeriphCLKConfig(&PeriphClkInit) != HAL\_OK)

{

\_Error\_Handler(\_\_FILE\_\_, \_\_LINE\_\_);

}

/\*\*Configure the Systick interrupt time

\*/

HAL\_SYSTICK\_Config(HAL\_RCC\_GetHCLKFreq()/1000);

/\*\*Configure the Systick

\*/

HAL\_SYSTICK\_CLKSourceConfig(SYSTICK\_CLKSOURCE\_HCLK);

/\* SysTick\_IRQn interrupt configuration \*/

HAL\_NVIC\_SetPriority(SysTick\_IRQn, 0, 0);

}

/\*\*

\* @brief NVIC Configuration.

\* @retval None

\*/

static void MX\_NVIC\_Init(void)

{

/\* TIM1\_UP\_IRQn interrupt configuration \*/

HAL\_NVIC\_SetPriority(TIM1\_UP\_IRQn, 0, 0);

HAL\_NVIC\_EnableIRQ(TIM1\_UP\_IRQn);

/\* ADC1\_2\_IRQn interrupt configuration \*/

HAL\_NVIC\_SetPriority(ADC1\_2\_IRQn, 0, 0);

HAL\_NVIC\_EnableIRQ(ADC1\_2\_IRQn);

/\* TIM3\_IRQn interrupt configuration \*/

HAL\_NVIC\_SetPriority(TIM3\_IRQn, 0, 0);

HAL\_NVIC\_EnableIRQ(TIM3\_IRQn);

}

/\* ADC1 init function \*/

static void MX\_ADC1\_Init(void)

{

ADC\_ChannelConfTypeDef sConfig;

/\*\*Common config

\*/

hadc1.Instance = ADC1;

hadc1.Init.ScanConvMode = ADC\_SCAN\_DISABLE;

hadc1.Init.ContinuousConvMode = DISABLE;

hadc1.Init.DiscontinuousConvMode = DISABLE;

hadc1.Init.ExternalTrigConv = ADC\_SOFTWARE\_START;

hadc1.Init.DataAlign = ADC\_DATAALIGN\_RIGHT;

hadc1.Init.NbrOfConversion = 1;

if (HAL\_ADC\_Init(&hadc1) != HAL\_OK)

{

\_Error\_Handler(\_\_FILE\_\_, \_\_LINE\_\_);

}

/\*\*Configure Regular Channel

\*/

sConfig.Channel = ADC\_CHANNEL\_0;

sConfig.Rank = ADC\_REGULAR\_RANK\_1;

sConfig.SamplingTime = ADC\_SAMPLETIME\_13CYCLES\_5;

if (HAL\_ADC\_ConfigChannel(&hadc1, &sConfig) != HAL\_OK)

{

\_Error\_Handler(\_\_FILE\_\_, \_\_LINE\_\_);

}

}

/\* TIM1 init function \*/

static void MX\_TIM1\_Init(void)

{

TIM\_ClockConfigTypeDef sClockSourceConfig;

TIM\_MasterConfigTypeDef sMasterConfig;

htim1.Instance = TIM1;

htim1.Init.Prescaler = 15;

htim1.Init.CounterMode = TIM\_COUNTERMODE\_UP;

htim1.Init.Period = 24323;

htim1.Init.ClockDivision = TIM\_CLOCKDIVISION\_DIV1;

htim1.Init.RepetitionCounter = 0;

htim1.Init.AutoReloadPreload = TIM\_AUTORELOAD\_PRELOAD\_DISABLE;

if (HAL\_TIM\_Base\_Init(&htim1) != HAL\_OK)

{

\_Error\_Handler(\_\_FILE\_\_, \_\_LINE\_\_);

}

sClockSourceConfig.ClockSource = TIM\_CLOCKSOURCE\_INTERNAL;

if (HAL\_TIM\_ConfigClockSource(&htim1, &sClockSourceConfig) != HAL\_OK)

{

\_Error\_Handler(\_\_FILE\_\_, \_\_LINE\_\_);

}

sMasterConfig.MasterOutputTrigger = TIM\_TRGO\_RESET;

sMasterConfig.MasterSlaveMode = TIM\_MASTERSLAVEMODE\_DISABLE;

if (HAL\_TIMEx\_MasterConfigSynchronization(&htim1, &sMasterConfig) != HAL\_OK)

{

\_Error\_Handler(\_\_FILE\_\_, \_\_LINE\_\_);

}

}

/\* TIM2 init function \*/

static void MX\_TIM2\_Init(void)

{

TIM\_MasterConfigTypeDef sMasterConfig;

TIM\_OC\_InitTypeDef sConfigOC;

htim2.Instance = TIM2;

htim2.Init.Prescaler = 2;

htim2.Init.CounterMode = TIM\_COUNTERMODE\_UP;

htim2.Init.Period = 4499;

htim2.Init.ClockDivision = TIM\_CLOCKDIVISION\_DIV1;

htim2.Init.AutoReloadPreload = TIM\_AUTORELOAD\_PRELOAD\_DISABLE;

if (HAL\_TIM\_PWM\_Init(&htim2) != HAL\_OK)

{

\_Error\_Handler(\_\_FILE\_\_, \_\_LINE\_\_);

}

sMasterConfig.MasterOutputTrigger = TIM\_TRGO\_RESET;

sMasterConfig.MasterSlaveMode = TIM\_MASTERSLAVEMODE\_DISABLE;

if (HAL\_TIMEx\_MasterConfigSynchronization(&htim2, &sMasterConfig) != HAL\_OK)

{

\_Error\_Handler(\_\_FILE\_\_, \_\_LINE\_\_);

}

sConfigOC.OCMode = TIM\_OCMODE\_PWM1;

sConfigOC.Pulse = 400;

sConfigOC.OCPolarity = TIM\_OCPOLARITY\_HIGH;

sConfigOC.OCFastMode = TIM\_OCFAST\_DISABLE;

if (HAL\_TIM\_PWM\_ConfigChannel(&htim2, &sConfigOC, TIM\_CHANNEL\_1) != HAL\_OK)

{

\_Error\_Handler(\_\_FILE\_\_, \_\_LINE\_\_);

}

HAL\_TIM\_MspPostInit(&htim2);

}

/\* TIM3 init function \*/

static void MX\_TIM3\_Init(void)

{

TIM\_ClockConfigTypeDef sClockSourceConfig;

TIM\_MasterConfigTypeDef sMasterConfig;

htim3.Instance = TIM3;

htim3.Init.Prescaler = 255;

htim3.Init.CounterMode = TIM\_COUNTERMODE\_UP;

htim3.Init.Period = 28124;

htim3.Init.ClockDivision = TIM\_CLOCKDIVISION\_DIV1;

htim3.Init.AutoReloadPreload = TIM\_AUTORELOAD\_PRELOAD\_DISABLE;

if (HAL\_TIM\_Base\_Init(&htim3) != HAL\_OK)

{

\_Error\_Handler(\_\_FILE\_\_, \_\_LINE\_\_);

}

sClockSourceConfig.ClockSource = TIM\_CLOCKSOURCE\_INTERNAL;

if (HAL\_TIM\_ConfigClockSource(&htim3, &sClockSourceConfig) != HAL\_OK)

{

\_Error\_Handler(\_\_FILE\_\_, \_\_LINE\_\_);

}

sMasterConfig.MasterOutputTrigger = TIM\_TRGO\_UPDATE;

sMasterConfig.MasterSlaveMode = TIM\_MASTERSLAVEMODE\_DISABLE;

if (HAL\_TIMEx\_MasterConfigSynchronization(&htim3, &sMasterConfig) != HAL\_OK)

{

\_Error\_Handler(\_\_FILE\_\_, \_\_LINE\_\_);

}

}

/\*\* Configure pins as

\* Analog

\* Input

\* Output

\* EVENT\_OUT

\* EXTI

\*/

static void MX\_GPIO\_Init(void)

{

GPIO\_InitTypeDef GPIO\_InitStruct;

/\* GPIO Ports Clock Enable \*/

\_\_HAL\_RCC\_GPIOC\_CLK\_ENABLE();

\_\_HAL\_RCC\_GPIOD\_CLK\_ENABLE();

\_\_HAL\_RCC\_GPIOA\_CLK\_ENABLE();

/\*Configure GPIO pin Output Level \*/

HAL\_GPIO\_WritePin(GPIOC, GPIO\_PIN\_13, GPIO\_PIN\_RESET);

/\*Configure GPIO pin Output Level \*/

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_12, GPIO\_PIN\_RESET);

/\*Configure GPIO pin : PC13 \*/

GPIO\_InitStruct.Pin = GPIO\_PIN\_13;

GPIO\_InitStruct.Mode = GPIO\_MODE\_OUTPUT\_PP;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_LOW;

HAL\_GPIO\_Init(GPIOC, &GPIO\_InitStruct);

/\*Configure GPIO pin : PA12 \*/

GPIO\_InitStruct.Pin = GPIO\_PIN\_12;

GPIO\_InitStruct.Mode = GPIO\_MODE\_OUTPUT\_PP;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_LOW;

HAL\_GPIO\_Init(GPIOA, &GPIO\_InitStruct);

}

/\* USER CODE BEGIN 4 \*/

void HAL\_TIM\_PeriodElapsedCallback(TIM\_HandleTypeDef \*htim){

HAL\_GPIO\_WritePin(GPIOA,GPIO\_PIN\_12,GPIO\_PIN\_SET);

if(htim->Instance == TIM1)

HAL\_ADC\_Start\_IT(&hadc1);

if(htim->Instance == TIM3){

if(setValue == 1363)

setValue = 2045;

else

setValue = 1363;

}

//HAL\_GPIO\_WritePin(GPIOA,GPIO\_PIN\_12,GPIO\_PIN\_SET);

}

void HAL\_ADC\_ConvCpltCallback(ADC\_HandleTypeDef\* hadc) {

rawValue = HAL\_ADC\_GetValue(&hadc1);

x[0] = (int32\_t)(setValue - ((4499\*rawValue)/4095));

y[0] = (2951\*x[0] - 4241\*x[1] + 1896\*x[2] + 761\*y[1] + 239\*y[2])/1000;

if(y[0] > 4499)

y[0] = 4499;

else if (y[0] < 0)

y[0] = 0;

\_\_HAL\_TIM\_SET\_COMPARE(&htim2, TIM\_CHANNEL\_1, (uint32\_t)y[0]);

y[2] = y[1];

y[1] = y[0];

x[2] = x[1];

x[1] = x[0];

HAL\_GPIO\_WritePin(GPIOA,GPIO\_PIN\_12,GPIO\_PIN\_RESET);

//HAL\_GPIO\_TogglePin(GPIOA,GPIO\_PIN\_12);

}

/\* USER CODE END 4 \*/

# BIBLIOGRAFIA

OGATA, Katsuhiko. Discrete-time control systmens. 2.ed. New Jersey: Prentice Hall, 1995. 745p., il. ISBN 0-13-034281-5.